



THE MC6809/MC6809E SYNC INSTRUCTION

In many applications, the microprocessor must be synchronized to external events. One possible solution is the use of external interrupts. Generally speaking, an interrupt causes the microprocessor to save (stack) all of its internal registers, fetch the interrupt vectors(s), and then jump to the interrupt routine. Processing then begins from that point. This is exactly what is done in the case of the MC6809/MC6809E microprocessor. However, the one problem encountered with interrupts is the latent period. For example, 18 cycles are required to perform the stacking operation for $\overline{\text{NMI}}$ or $\overline{\text{IRQ}}$ and nine cycles are required to perform the stacking operation for $\overline{\text{FIRQ}}$. Also, the current instruction must be completed before the stacking operation can be started. As an example, if a hardware interrupt is recognized during the first machine cycle of an SWI instruction, which requires 19 cycles to execute, 27 cycles would elapse before the interrupt service routine could begin, provided it was an $\overline{\text{FIRQ}}$ interrupt (for $\overline{\text{NMI}}$ or $\overline{\text{IRQ}}$ the time would be nine cycles longer). If an $\overline{\text{FIRQ}}$ interrupt is recognized during the last machine cycle of any instruction, the response time would be nine cycles (again the time would be correspondingly longer for an $\overline{\text{NMI}}$ or $\overline{\text{IRQ}}$). In many cases, the resulting latent period would be unacceptable.

An example of the effects produced by the above discussion are examined in the following case:

- The MPU is controlling a serial transmission port using an MC6854 advanced data link controller (ADLC).
- The serial transmission rate is 1 MHz.
- The ADLC uses synchronous data link control (SDLC) format.
- The first eight bits of the incoming message following the opening flag will be an 8-bit address byte.
- An address present interrupt is generated.

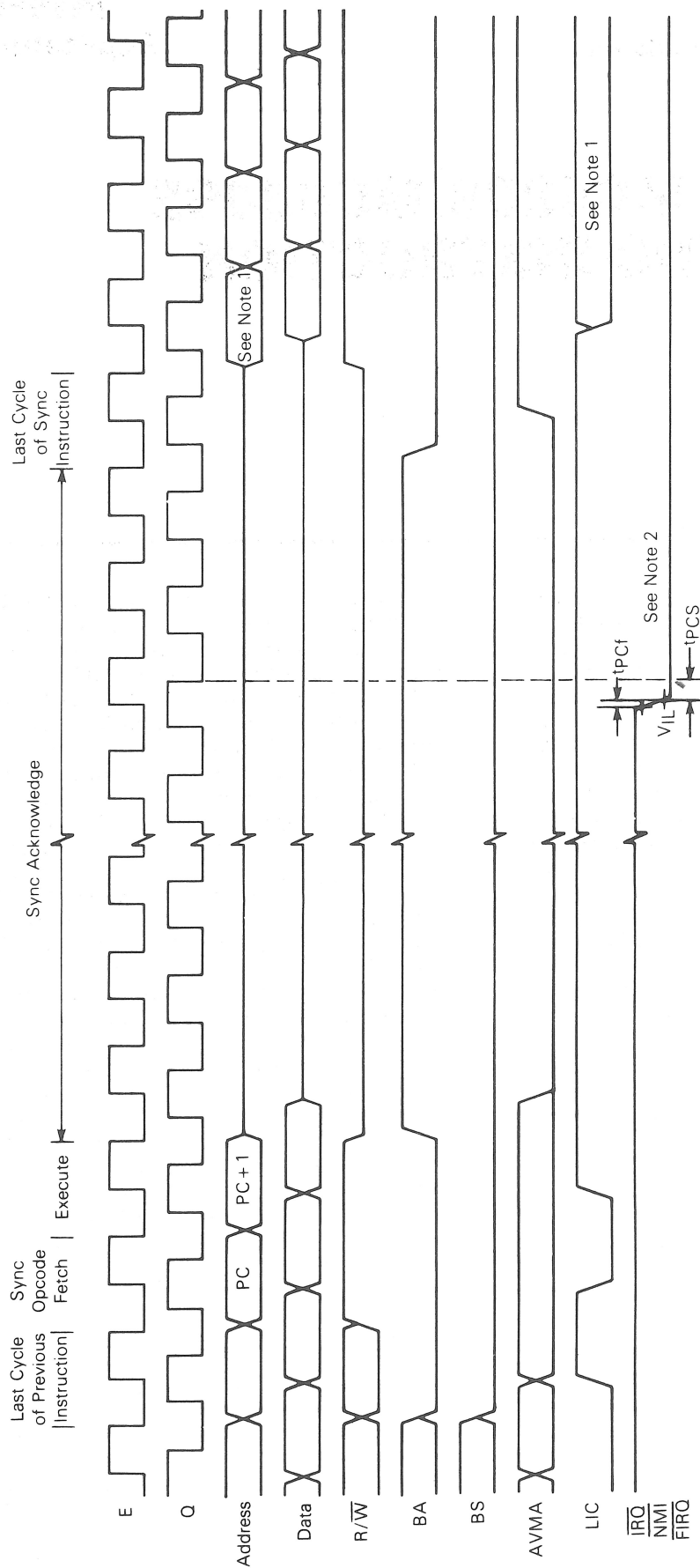
Under these circumstances, the MPU has eight microseconds to sample the address byte to determine if the message is for this station. Even if $\overline{\text{FIRQ}}$ were used, the address byte would be lost (overwritten by the next byte) before an interrupt routine could be entered to examine it. (Application Note AN-830 "An Intelligent Terminal with Data Link

Capability" gives a detailed explanation of SDLC and the MC6854 and the programs and protocols used.) A solution for the above problem is available by using the SYNC instruction of the MC6809/MC6809E MPU.

The SYNC instruction of the MC6809/MC6809E MPU causes the MPU to enter a synchronizing state and the processor sits in a waiting loop until an interrupt occurs. The address bus is in a high-impedance state (as opposed to \$FFFF), the data bus is in a don't care condition, and BA and BS equal 1 and 0, respectively. The SYNC instruction will be exited upon the occurrence of any hardware interrupt.

If the pending interrupt is non-maskable ($\overline{\text{NMI}}$) or maskable ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) with its mask bit (F or I) clear, the processor will then clear the SYNC state and perform the normal interrupt stacking and service routine. Since $\overline{\text{FIRQ}}$ and $\overline{\text{IRQ}}$ are not edge triggered, a low level with a minimum duration of three machine cycles is required to ensure that the interrupt will be recognized. The only advantage of being in a SYNC state with the mask bits clear is that only one machine cycle will occur before the stacking operation occurs in response to an interrupt instead of completing an instruction. If a maskable interrupt ($\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$) occurs during a SYNC instruction while the respective bit is set, the processor will exit the SYNC instruction and continue by executing the next instruction following the SYNC instruction. The SYNC instruction timing is shown in Figure 1. No stacking of registers will occur for this case. Thus, the latent period is reduced to one machine cycle.

As pointed out in the above explanation, the SYNC instruction offers no advantage over a wait loop if the interrupt is **enabled**. However, if the interrupt is masked, only one machine cycle will occur between recognition of the interrupt and the execution of the first instruction following the SYNC instruction. Now consider the two programs shown in Figure 2. Program A is a wait loop followed by the service routine to examine the address word of the ADLC. Program B contains a routine which does the same function using the SYNC instruction. Program A cannot execute the LDA instruction for less than nine cycles and a maximum of 11 cycles. A branch instruction requires three cycles to execute.



- Notes:
1. If the associated mask bit is set when the interrupt is requested, LIC will go low and this cycle will be an instruction fetch from address location PC + 1. However, if the interrupt is accepted (NMI or an unmasked FIRQ or IRQ) LIC will remain high and interrupt processing will start with this cycle as (m) on Figures 9 and 10 (Interrupt Timing).
 2. If mask bits are clear, IRQ and FIRQ must be held low for three cycles to guarantee that interrupt will be taken, although only one cycle is necessary to bring the processor out of SYNC.

NOTE: timing measurements are referenced to and from a low voltage of 0.8 volt and a high voltage of 2.0 volts, unless otherwise noted.

FIGURE 1 — SYNC Instruction Timing Diagram

Since data is not fetched until the fifth cycle of an LDA instruction, 14 to 16 cycles occur between the address present interrupt and the attempted fetch of the address byte. However, program B will allow the LDA instruction to be executed one cycle after the interrupt and the fetch of the address byte will occur six cycles after the interrupt. Because the interrupt cannot be asserted until the machine cycle following the actual setting of the address present bit, seven cycles can actually occur from interrupt recognition until the address byte is fetched.

From the above discussion, the primary function of the SYNC instruction is to ensure that the response to an interrupt occurs rapidly. There are other uses limited only by the designer's imagination. Consider the example of data being shifted into a shift register in 16-bit words and then read by two different MPUs. This system, which is shown in Figure 3, might represent data received from a radar set. The first eight bits might represent azimuth and the second eight

bits might represent range. Each MPU must know when to read its data. This can be accomplished by synchronizing the programs using the SYNC instruction. On each 16th bit, an interrupt is fed to $\overline{IRQ}$ and both processors fetch and digest the data gathered from the shift register. Upon completion of their programs, both processors will execute a SYNC instruction and wait for the next interrupt which indicates that new data is available. This example assumes that the interrupts are masked so that no stacking occurs.

Many cases exist which are time-critical applications. Many of these cases are associated with serial data in various forms where normal interrupt routines cannot respond. By using the SYNC instruction rather than a wait loop to be excited by an interrupt, response time of the MPU can be shortened sufficiently to eliminate the need for extra hardware holding registers or other external circuitry. With new instructions such as SYNC, the performance of MPUs continues to be expanded.

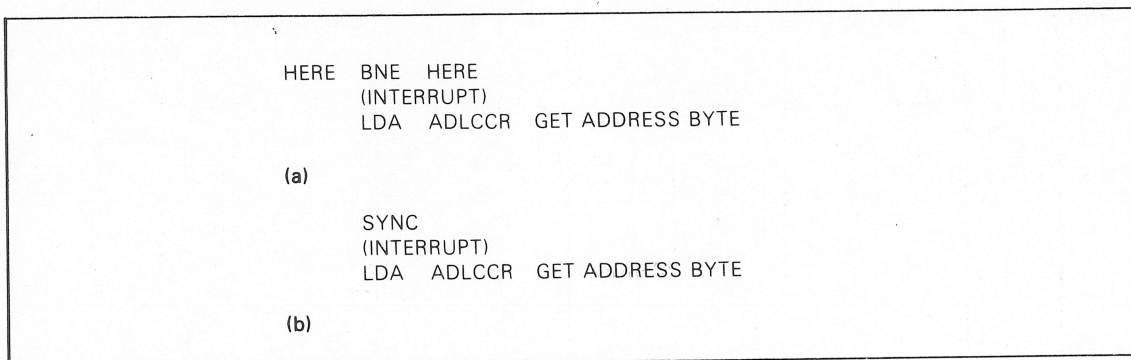


FIGURE 2 — Programs Using Wait Loop and SYNC Instruction

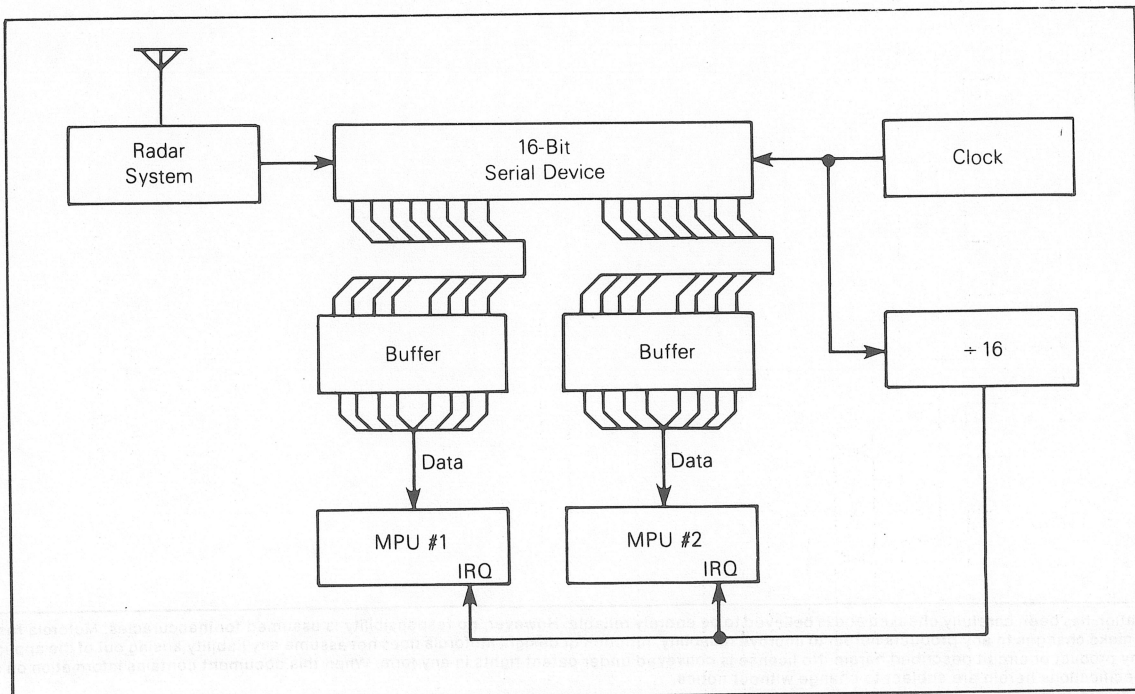


FIGURE 3 — Hypothetical Radar Unit Using Two MPUs

This information has been carefully checked and is believed to be entirely reliable. However, no responsibility is assumed for inaccuracies. Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein. No license is conveyed under patent rights in any form. When this document contains information on a new product, specifications herein are subject to change without notice.



MOTOROLA Semiconductor Products Inc.

Colvilles Road, Kelvin Estate - East Kilbride/Glasgow - SCOTLAND

Printed in Switzerland